What is Claimed Is:

1. A system for processing network data packets using a hardware engine, comprising:

a TCP Table manager for managing a TCP connection's state information by providing a pool of buffers used for data structures and providing plural registers and timer functions to other system sub-modules.

2. The system of Claim 1, wherein the TCP Table manager maintains a free list of data structures that are used for storage of TCP connection state and storage of TCP transfer requests.

3. The system of Claim 1, wherein the TCP Table Manager provides a cache of network control blocks ("NCBs").

4. The system of Claim 1, wherein the TCP Table Manager has a command processor that arbitrates between plural command sources and translates a received command to an output action(s) to other TCP Table Manager components.

5. The system of Claim 4, wherein the TCP table manager's command processor co-ordinates inbound and/or outbound channel access to network data structures.

6. The system of Claim 1, wherein the TCP Table Manager provides a read / write register interface to allow simultaneous access to fields within the NCBs.

7. The system of Claim 6, wherein the TCP table manager's register interface provides a locking mechanism to allow sole access to a particular field within an NCB.

8. The system of Claim 1, wherein the TCP table manager includes a timer list manager that maintains timer functions for all TCP connections.

9. The of Claim 8, wherein the timer list manager maintains a persist timer, an idle timer, a delayed ACK timer and/or a retransmit timer for each TCP connection.

10. The system of Claim 8, wherein the timer list manager scans a timer list and checks for timed events that have expired.

11. The system of Claim 1, wherein the TCP table manager maintains an outbound request list to signal if a network control block is ready for transmit processing due to a timer event or a receive event.

12. A system for processing network data packets using a hardware engine, comprising:

an outbound TCP processor that takes requests from a host to transmit TCP data, transmits the TCP data following TCP rules and signals to a host when the transmission is complete and has arrived on the remote node; and transmits TCP acknowledgements in response to TCP data received.

13. The system of Claim 12, further comprising:

a request manager that downloads an input/output control block ("IOCB") and determines what action is required with respect to the downloaded IOCB.

14. The system of Claim 12, further comprising:

a window update module that processes plural events from a TCP Table Manager and an Inbound TCP Processor.

15. The system of Claim 12, further comprising:

a main control module that determines if a network control block is in a valid state to send data and how much data needs to be sent.

16. The system of Claim 12, further comprising:

a completion module that sends completion messages when the outbound TCP processor completes processing a command and/or operation.

17. The system of Claim 12, further comprising:

an IP Interface module that signals an IP processor module to build an IP and MAC protocol header and then builds a TCP header based on information obtained from a network control block ("NCB").

18. The system of Claim 12, further comprising:

an outbound DMA engine interface that scans a scatter/gather list of buffers to be transmitted, fetches the proper length of data to be sent and passes this data to the IP processor module to be concatenated with built TCP/IP headers.

19. A system for processing network data packets using a hardware engine, comprising:

an inbound IP fragment processor that receives IP datagram fragments and manages the reassembly of any number of in-process datagrams, wherein re-assembled datagrams are passed to a TCP processor, or to a host for non-TCP packets.

20. The system of Claim 19, further comprising:

an input processor for parsing data packet header information, assembling datagrams, and interfacing with an output processor and a return processor.

21. The system of Claim 20, wherein if a datagram is not complete, the input processor verifies if an entry exists in a reassembly list and adds an entry if no entry exists.

22. The system of Claim 20, wherein the input processor sets a first fragment flag when a datagram is added to the reassembly list.

23. The system of Claim 20, wherein if an entry already exists and the datagram is found in the reassembly list, then the fragment is added to the reassembly list for the datagram.

24. The system of Claim 20, wherein when a fragment is added to the reassembly list, the input processor verifies if an incoming fragment is sequential to either a previous or next fragment and trims the fragment if sequential.

25. The system of Claim 20, wherein if a reassembled datagram is not destined for a TCP address, then it is sent to a host system for disposition.

26. The system of Claim 20, wherein the input processor passes a complete datagram to an output processor.

27. The system of Claim 20, wherein if a reassembled datagram is destined for a TCP address, then the datagram is passed to the output processor.

28. The system of Claim 20, wherein the output processor parses fully reassembled IP datagrams that are destined for TCP, to a TCP processor.

29. The system of Claim 19, further comprising:

a return processor that takes the buffers used to receive IP fragments and returning the buffers to a free buffer pool if an error occurred in the reassembly of the datagram or if the fragment was not necessary for the reassembly.